Project 3 – Evaluation of IR Model

Prakhathi Murugesan – prakhath (50316846)

Abstract

The purpose of this project is to implement different IR models, evaluate them and improve the search results by modifying the various parameters. Implementation of IR models such as BM25(Best Matching) model, DFR (Divergence from Randomness) models and Language model based on Solr using twitter data has been done. The results obtained from each of these models is taken for Trec Eval, where the results are compared against the ground truth judgement values given by experts. The MAP(Mean Average Precision) value, from Trec-eval is used for comparing the performance of the models.

1. Introduction:

The goal is to implement IR models and evaluate them. The IR models used are BM25(Best Matching) model, DFR (Divergence from Randomness) models and Language model. Training tweet data is used as the dataset which has three languages include – English(text_en), German(text_de) and Russian(text_ru). Dataset contains approximately 3500 tweets.

1.1 BM 25 Model:

Okapi BM25 model is a probabilistic information retrieval model which was originally designed for short-length documents. The function ranks a set of documents based on the query terms appearing in each document. In Solr, the similarity class for this is solr.BM25SimilarityFactory.

The retrieval status value for BM25 is calculated using

$$RSV_d = \sum_{t \epsilon q} \log\left[\frac{N}{df_t}\right] * \frac{(k_1+1)tf_{td}}{k_1\left((1-b)+b*\left(\frac{L_d}{L_{avg}}\right)\right)+\ tf_{td}}$$

The following snippet is used to configure BM25 model in solr:

```
<similarity class="solr.BM25SimilarityFactory">
    <str name="b">0.75</str>
    <str name="k1">1.2</str>
  </similarity>
```

1.2 Divergence from Randomness model (DFR):

In Divergence from randomness model, the term-weight is inversely related to the probability of term-frequency within the document obtained by a model of randomness. These models are term-document matching functions that are obtained by the product of two divergence functions. In Solr, the similarity class for this is given by solr.DFRSimilarityFactory.

The following snippet is used to configure DFR model in solr:

```
<similarity class="solr.DFRSimilarityFactory">
  <str name="basicModel">G</str>
  <str name="afterEffect">B</str>
  <str name="normalization">H2</str>
  <str name="c">7.0</str>
</similarity>
```

1.3 Language Model:

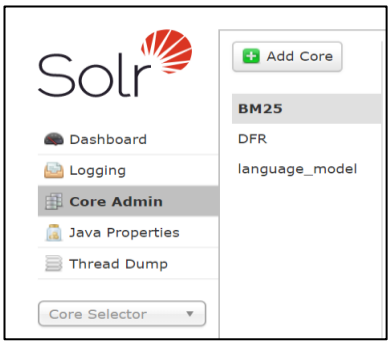
The goal of a language model is to assign a probability to a sequence of words by means of a probability distribution. The model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability to the whole sequence. The language model provides context to distinguish between words and phrases that sound similar. In Solr, the similarity class for this is given by LMDirichletSimilarityFactory.

The following snippet is used to configure Language model in solr:

```
<similarity class="solr.LMDirichletSimilarityFactory">
  <float name="mu">2500</float>
</similarity>
```

2. Setting up the solr:

To implement three different IR model, we need to create three cores in solr. These cores are created with different schema and solrconfig files. The tweet data is indexed on all the three cores and query is executed for each model and scores for each document is retrieved.



To improve the performance, instead of standard query parser we use customised query parsers called Extended query parser(edismax). This is configured in solrconfig file and defType is used to define this parser and it is used while querying. In addition to query parser, we use query boosting to improve the scores.

2.1 Extended DisMax query parser:

The Extended DisMax (eDisMax) query parser is an improved version of the DisMax query parser. In addition to supporting all the DisMax query parser parameters. eDisMax provides support to search across

multiple fields. It searches the query terms across every specified field in the '**qf**' parameter. Edismax can run queries against all query fields and phrase fields. The phase query can have slop which is the distance between the terms of the query while still considering it a phrase match.

The Extended DisMax includes the following query parameters:

- **The 'ps' Paremeter** – Default amount of slop on phrase queries built with pf, pf2, and/or pf3 fields.
- **The 'qf' parameter** - A list of fields, each of which is assigned a boost factor to increase or decrease that particular field's importance in the query.
- **The 'pf' parameter -** boosts the score of documents in cases where all of the terms in the q parameter appear in close proximity.
- **The 'pf2' Parameter** - A list of fields with optional weights, based on pairs of word shingles.
- **The 'ps2' Parameter** - This is similar to ps but overrides the slop factor used for pf2. If not specified, ps is used.
- **The 'pf3' Parameter** - A list of fields with optional weights, based on triplets of word shingles. Similar to pf, except that instead of building a phrase per field out of all the words in the input, it builds a set of phrases for each field out of each triplet of word shingles.
- **The 'ps3' Parameter** - This is similar to ps but overrides the slop factor used for pf3. If not specified, ps is used.

The following snippet is used to configure edismax query parser in solr:

```
<str name="defType">edismax</str>
<str name="qf">text_en^3 text_ru^3 text_de^3</str>
<int name="qs">10</int>
<str name="pf">text_en^2 text_ru^2 text_de^2</str>
<int name="ps">10</int>
<str name="pf2">text_en^10 text_ru^10 text_de^10</str>
<int name="ps2">5</int>
```

2.2 Boost Query

The Boost Query parameter specifies an additional query clause that will be added to the user's main query to influence the score. Multiple Boost Query parameters can be used if a query has to be parsed as separate clauses with separate boosts.

The below snippet is used to boost the score if the query is present in any of the three fields.

```
<str name="pf">text_en^2 text_ru^2 text_de^2</str>
```

3. Tuning the parameters values

The tuning process involves the tweaking of parameters of IR models in order to improve the performance of the models.

3.1 Tuning B and K1 values in BM25:

Parameters:

- K1 (float): Controls non-linear term frequency normalization (saturation). The default is 1.2
- b (float): Controls to what degree document length normalizes tf values. The default is 0.75

For the default setting, MAP = 0.67. To obtain good MAP, the values of K1 and b has been tuned between 0 to 3 and 0 to 1 respectively. The below table represents the respective b, K1 and MAP values.

b	K1	MAP
0.95	1.75	0.698
0.92	1.2	0.7022
0.75	1.2	0.7227
0.85	1	0.7234
0.95	0.95	0.7237
0.5	0.5	0.7239
0.75	0.75	0.7254
0.85	0.85	0.7265

3.2 Tuning mu value in Language model:

Parameters:

- Parameter mu (float): smoothing parameter μ. The default is 2000

To obtain good MAP, the value of mu has been tuned. The below table represents the respective mu and MAP values.

mu	MAP
3000	0.6808
2000	0.6835
1500	0.6846
500	0.7067
100	0.7093

3.3 Tuning parameter values in DFR model:

Parameters:

- G: Geometric approximation of Bose-Einstein
- B: Ratio of two Bernoulli processes
- H2: term frequency density inversely related to length
  - Parameter c (float): hyper-parameter that controls the term frequency normalization with respect to the document length. The default is 1

By modifying the values of G, B and H2, desired results to obtain good MAP can be achieved. The below table represents the respective values.

Basic Model	After Effect	Normalization	MAP
I	B	H1	0.2444
G	B	H2	0.251

After adding edismax query parser,

Basic Model	After Effect	Normalization	PS	PS2	PS3	MAP
G	B	H2	5	-	-	0.7026
G	B	H2	10	5	-	0.7169
G	B	H2	10	5	10	0.7356

4. Evaluation Model

To evaluate the performance of the various model, we use trev_eval program. The results obtained for each query contains similarity score which is used by trev_eval to evaluate the performance of the model. It uses the relevance value or ground truth judgement value for each query and compares it with result and gives out the output. Among the various values, we use MAP value to compare the performance of various model.

4.1 MAP score

The mean average precision (MAP) of a set of queries is defined by

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

where *Q* is the number of queries in the set and AveP(q) is the average precision (AP) for a given query, q.

For a given query, q, its corresponding AP is calculated, and then the mean of all these AP scores would give a single number, called MAP, which quantifies how good the model is at performing the query.

MAP value obtained for DFR model:

```
C:\Users\prakhathi\Documents\IR\project3_data\project3_data>trec_eval -q -c -M 1000 -m map qrel.txt DFRtest.txt
map                     001     0.4798
map                     002     0.6419
map                     003     0.8693
map                     004     0.7233
map                     005     0.5000
map                     006     0.5281
map                     007     1.0000
map                     008     1.0000
map                     009     1.0000
map                     010     1.0000
map                     011     0.9861
map                     012     0.6598
map                     013     0.1365
map                     014     0.7373
map                     015     0.7721
map                     all     0.7356
```

MAP value obtained for BM25 model:

```
C:\Users\prakhathi\Documents\IR\project3_data\project3_data>trec_eval -q -c -M 1000 -m map qrel.txt BM25-mod.txt
map                     001     0.6164
map                     002     0.6016
map                     003     0.8544
map                     004     0.5531
map                     005     0.6875
map                     006     0.5245
map                     007     1.0000
map                     008     1.0000
map                     009     1.0000
map                     010     1.0000
map                     011     0.9861
map                     012     0.4291
map                     013     0.1361
map                     014     0.7373
map                     015     0.7721
map                     all     0.7265
```

MAP value obtained for Language model:

```
C:\Users\prakhathi\Documents\IR\project3_data\project3_data>trec_eval -q -c -M 1000 -m map qrel.txt language_model-mod.t
xt
map                     001     0.4315
map                     002     0.6541
map                     003     0.7500
map                     004     0.7264
map                     005     0.5375
map                     006     0.5361
map                     007     0.7500
map                     008     1.0000
map                     009     1.0000
map                     010     1.0000
map                     011     0.9861
map                     012     0.6302
map                     013     0.1286
map                     014     0.7373
map                     015     0.7721
map                     all     0.7093
```

Since we have provided 15 queries, the MAP value for each query is obtained and the overall MAP value for each model is obtained. And the observation made is that, DFR model gives the better performance with MAP value of 0.7356.

5. Result:

From the above experiment and observation, it is clear that the DFR model gives the best performance with MAP value of 0.7356 when we implement the model with below specifications:

Basic Model	G
After Effect	B
Normalization	H2
PS	10
PS2	5
PS3	10

Reference:

1. https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-TheboostParameter
2. https://blog.thedigitalgroup.com/understanding-phrasequery-and-slop-in-solr